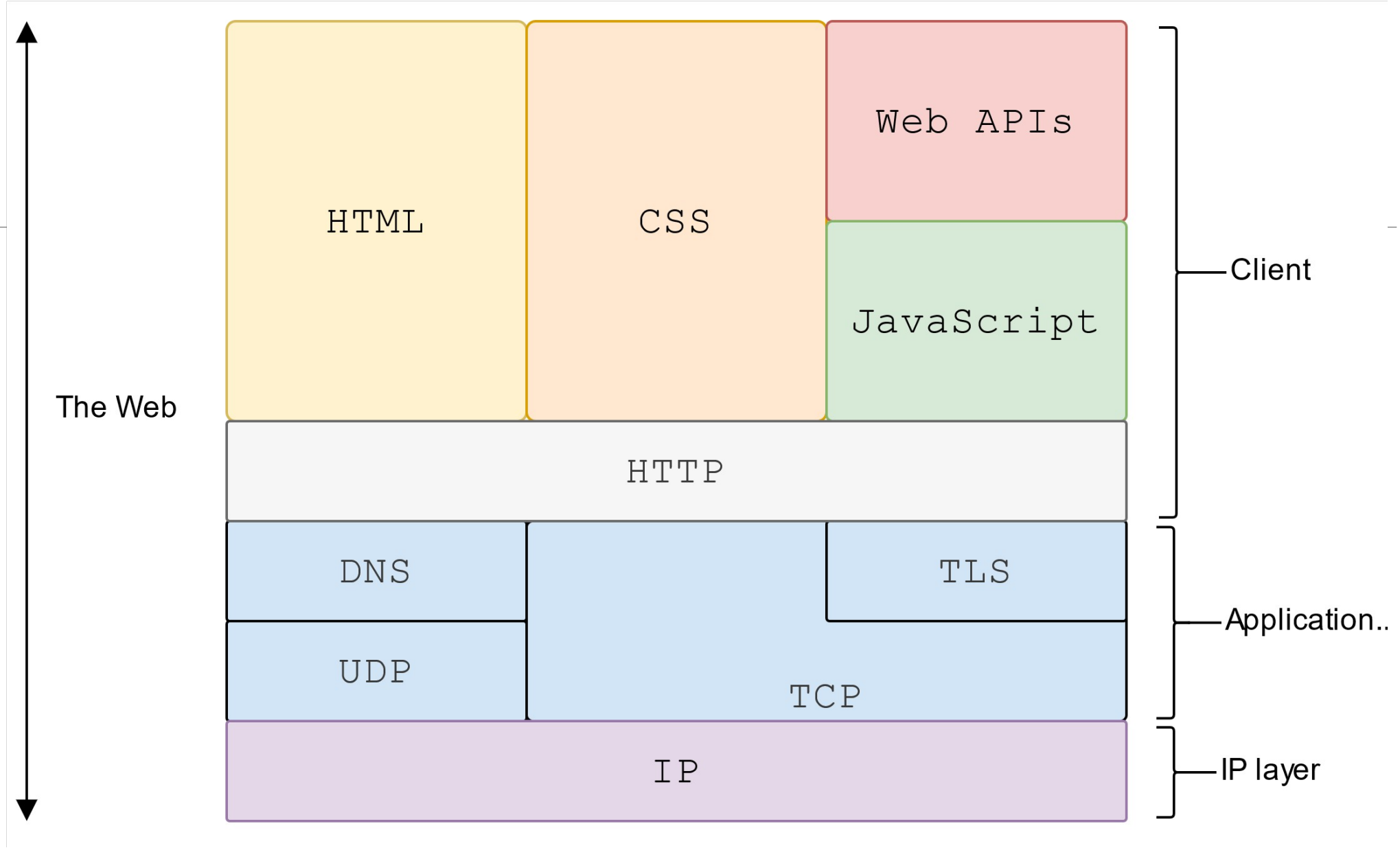
HTTP Basics and Headers

UNIT 1- WEB PEN TESTING- SESSIONS 1-2

What is HTTP?

1. **HTTP** is a protocol or fetching resources such as HTML documents. It is the foundation of any data exchange on the Web and it is a client-server protocol, which means requests are initiated by the recipient, usually the Web browser. A complete document is typically constructed from resources such as text content, layout instructions, images, videos, scripts, and more.
2. Clients and servers communicate by exchanging individual messages (as opposed to a stream of data). The messages sent by the client are called *requests* and the messages sent by the server as an answer are called *responses*.
3. Designed in the early 1990s, HTTP is an extensible protocol which has evolved over time. It is an application layer protocol that is sent over TCP, or over a TLS-encrypted TCP connection, though any reliable transport protocol could theoretically be used. Due to its extensibility, it is used to not only fetch hypertext documents, but also images and videos or to post content to servers, like with HTML form results. HTTP can also be used to fetch parts of documents to update Web pages on demand.


The Web
HTML
CSS
Web APIs
JavaScript
HTTP
DNS
TLS
UDP
TCP
IP
Client
Application..
IP layer

Components of HTTP Systems

1. Client: Web Browser
2. Web Server
3. Proxies

Proxies

HTTP is a client-server protocol: requests are sent by one entity, the user-agent (or a proxy on behalf of it). Most of the time the user-agent is a Web browser, but it can be anything, for example, a robot that crawls the Web to populate and maintain a search engine index.

Each individual request is sent to a server, which handles it and provides an answer called the *response*. Between the client and the server there are numerous entities, collectively called proxies, which perform different operations and act as gateways or caches.

Client: Web browser

1. The *user-agent* is any tool that acts on behalf of the user. This role is primarily performed by the Web browser, but it may also be performed by programs used by engineers and Web developers to debug their applications.
2. The browser is **always** the entity initiating the request. It is never the server (though some mechanisms have been added over the years to simulate server-initiated messages).
3. To display a Web page, the browser sends an original request to fetch the HTML document that represents the page. It then parses this file, making additional requests corresponding to execution scripts, layout information (CSS) to display, and sub-resources contained within the page (usually images and videos). The Web browser then combines these resources to present the complete document, the Web page. Scripts executed by the browser can fetch more resources in later phases and the browser updates the Web page accordingly.
4. A Web page is a hypertext document. This means some parts of the displayed content are links, which can be activated (usually by a click of the mouse) to fetch a new Web page, allowing the user to direct their user-agent and navigate through the Web. The browser translates these directions into HTTP requests, and further interprets the HTTP responses to present the user with a clear response.

Web Server

1. On the opposite side of the communication channel is the server, which *serves* the document as requested by the client. A server appears as only a single machine virtually; but it may actually be a collection of servers sharing the load (load balancing), or other software (such as caches, a database server, or e-commerce servers), totally or partially generating the document on demand.
2. A server is not necessarily a single machine, but several server software instances can be hosted on the same machine. With HTTP/1.1 and the Host header, they may even share the same IP address.

Character of HTTP

HTTP is simple

HTTP is generally designed to be human-readable, even with the added complexity introduced in HTTP/2 by encapsulating HTTP messages into frames. HTTP messages can be read and understood by humans, providing easier testing for developers, and reduced complexity for newcomers.

HTTP is extensible

Introduced in HTTP/1.0, HTTP headers make this protocol easy to extend and experiment with. New functionality can even be introduced by an agreement between a client and a server about a new header's semanti

HTTP is stateless, but not sessionless

HTTP is stateless: there is no link between two requests being successively carried out on the same connection. This immediately has the prospect of being problematic for users attempting to interact with certain pages coherently, for example, using e-commerce shopping baskets. But while the core of HTTP itself is stateless, HTTP cookies allow the use of stateful sessions. Using header extensibility, HTTP Cookies are added to the workflow, allowing session creation on each HTTP request to share the same context, or the same state.

HTTP and connections

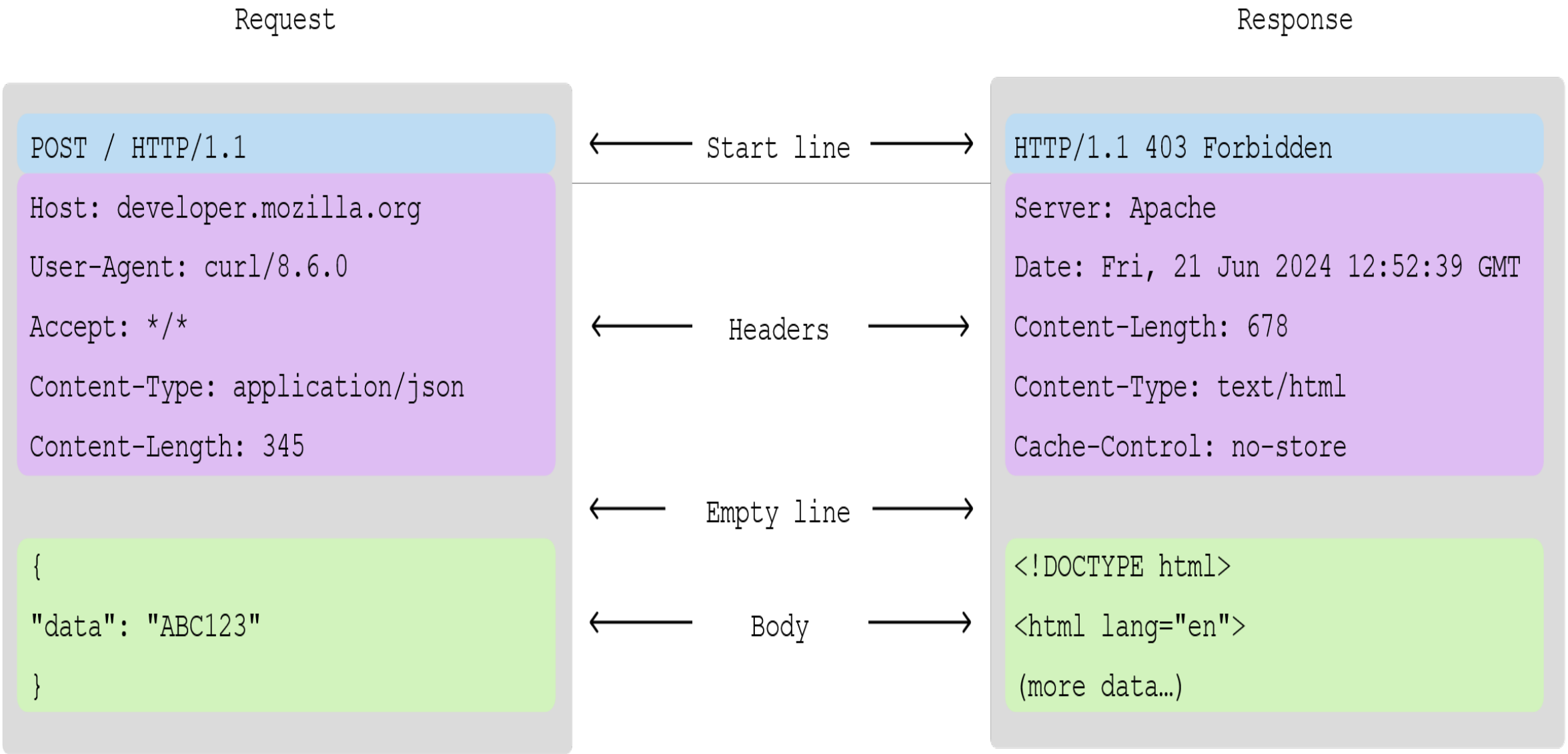
A connection is controlled at the transport layer, and therefore fundamentally out of scope for HTTP. HTTP doesn't require the underlying transport protocol to be connection-based; it only requires it to be *reliable*, or not lose messages (at minimum, presenting an error in such cases). Among the two most common transport protocols on the Internet, TCP is reliable and UDP isn't. HTTP therefore relies on the TCP standard, which is connection-based.

Flow of HTTP

1. Open a TCP connection: The TCP connection is used to send a request, or several, and receive an answer. The client may open a new connection, reuse an existing connection, or open several TCP connections to the servers.
2. Send an HTTP message: HTTP messages (before HTTP/2) are human-readable. With HTTP/2, these messages are encapsulated in frames, making them impossible to read directly, but the principle remains the same.
3. Read the response and close the connection

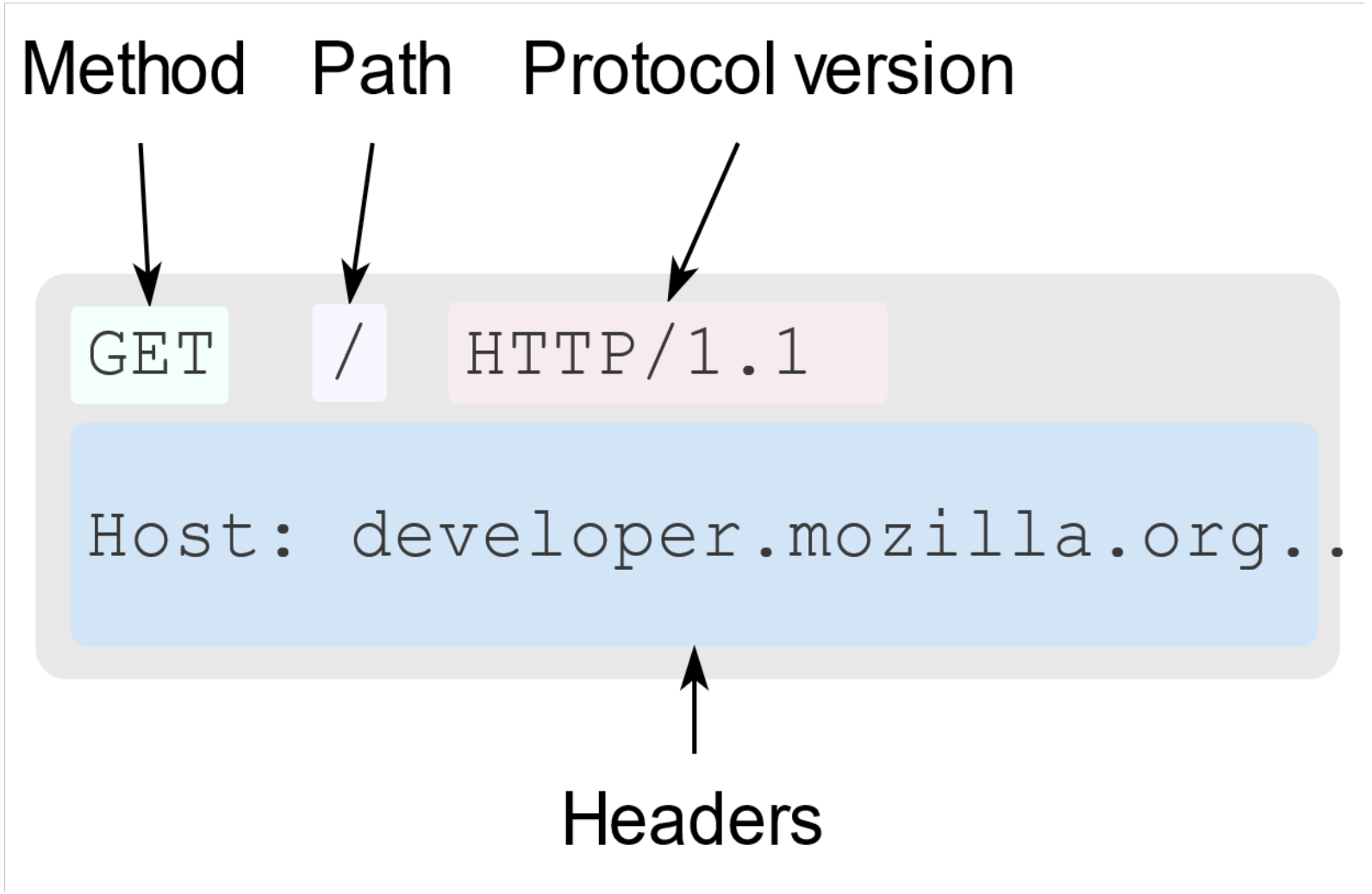
HTTP Messages

1. HTTP messages, as defined in HTTP/1.1 and earlier, are human-readable. In HTTP/2, these messages are embedded into a binary structure, a *frame*, allowing optimizations like compression of headers and multiplexing. Even if only part of the original HTTP message is sent in this version of HTTP, the semantics of each message is unchanged and the client reconstitutes (virtually) the original HTTP/1.1 request. It is therefore useful to comprehend HTTP/2 messages in the HTTP/1.1 format.
2. There are two types of HTTP messages, requests and responses, each with its own format.


Request
Response
POST / HTTP/1.1
Host: developer.mozilla.org
User-Agent: curl/8.6.0
Accept: */*
Content-Type: application/json
Content-Length: 345
{
"data": "ABC123"
}
Start line
Headers
Empty line
Body
HTTP/1.1 403 Forbidden
Server: Apache
Date: Fri, 21 Jun 2024 12:52:39 GMT
Content-Length: 678
Content-Type: text/html
Cache-Control: no-store
<!DOCTYPE html>
<html lang="en">
(more data…)

HTTP

•An HTTP method, usually a verb like GET, POST, or a noun like OPTIONS or HEAD that defines the operation the client wants to perform. Typically, a client wants to fetch a resource (using GET) or post the value of an HTML form (using POST), though more operations may be needed in other cases.
•The path of the resource to fetch; the URL of the resource stripped from elements that are obvious from the context, for example without the protocol (http://), the domain (here, developer.mozilla.org), or the TCP port (here, 80).
•The version of the HTTP protocol.
•Optional headers that convey additional information for the servers.
•A body, for some methods like POST, similar to those in responses, which contain the resource sent.



HTTP Request Methods

GET
The GET method requests a representation of the specified resource. Requests using GET should only retrieve data and should not contain request content.

HEAD
The HEAD method asks for a response identical to a GET request, but without a response body.

POST
The POST method submits an entity to the specified resource, often causing a change in state or side effects on the server.

PUT
The PUT method replaces all current representations of the target resource with the request content.

DELETE
The DELETE method deletes the specified resource.

CONNECT
The CONNECT method establishes a tunnel to the server identified by the target resource.

OPTIONS
The OPTIONS method describes the communication options for the target resource.

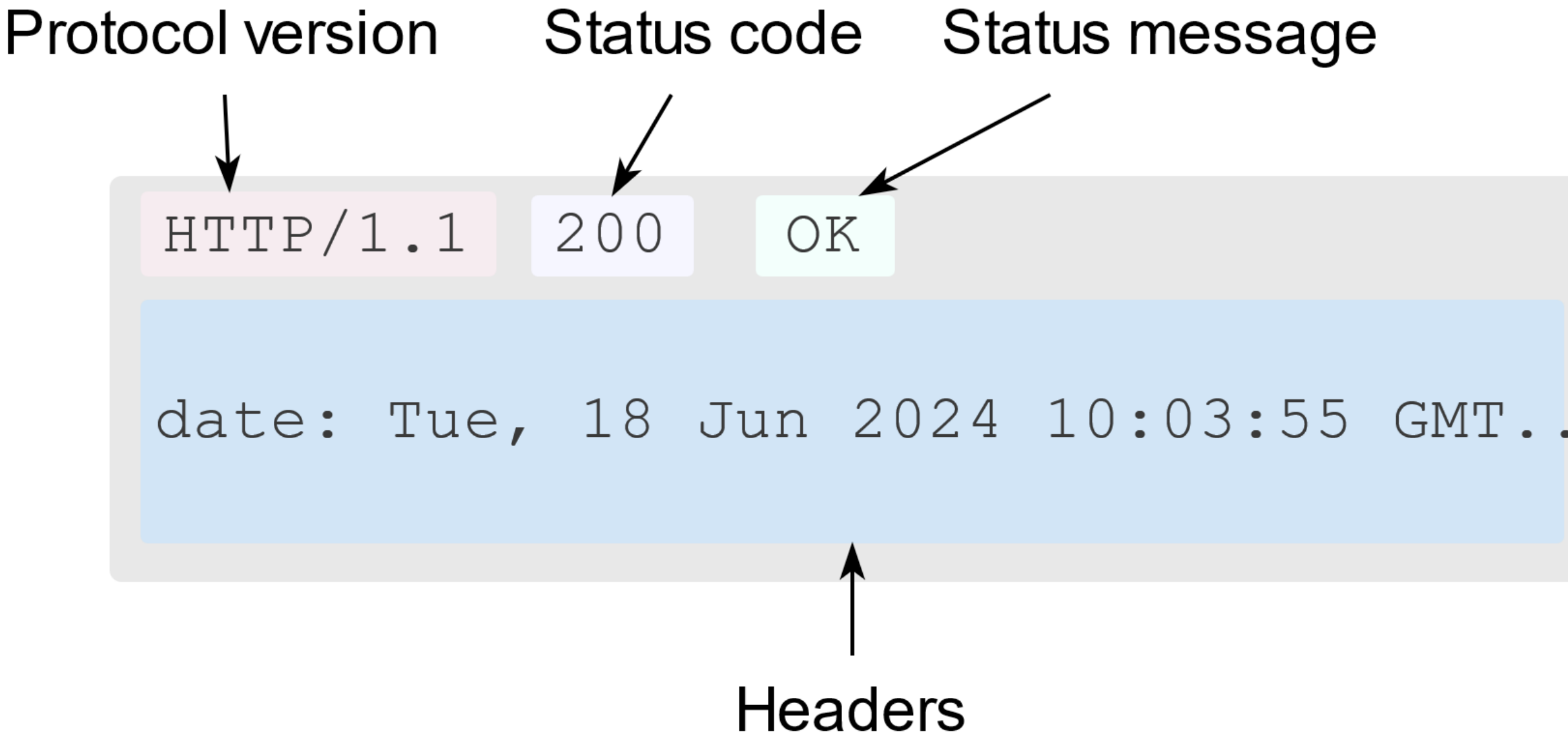
TRACE
The TRACE method performs a message loop-back test along the path to the target resource.

PATCH
The PATCH method applies partial modifications to a resource.

HTTP Response

Responses consist of the following elements:

- The version of the HTTP protocol they follow.
- A status code, indicating if the request was successful or not, and why.
- A status message, a non-authoritative short description of the status code.
- HTTP headers, like those for requests.
- Optionally, a body containing the fetched resource.


Protocol version
Status code
Status message
HTTP/1.1 200 OK
date: Tue, 18 Jun 2024 10:03:55 GMT..
Headers

HTTP Response Status Codes

HTTP response status codes indicate whether a specific HTTP request has been successfully completed. Responses are grouped in five classes:

1. Informational responses (100 – 199)
2. Successful responses (200 – 299)
3. Redirection messages (300 – 399)
4. Client error responses (400 – 499)
5. Server error responses (500 – 599)

200 OK

The request succeeded, and a new resource was created as a result. This is typically the response sent after POST requests, or some PUT requests.

202 Accepted

The request has been received but not yet acted upon. It is noncommittal, since there is no way in HTTP to later send an asynchronous response indicating the outcome of the request. It is intended for cases where another process or server handles the request, or for batch processing.

301 Moved Permanently

The URL of the requested resource has been changed permanently. The new URL is given in the response.

302 Found

This response code means that the URI of requested resource has been changed *temporarily*. Further changes in the URI might be made in the future, so the same URI should be used by the client in future requests.

400 Bad Request

The server cannot or will not process the request due to something that is perceived to be a client error (e.g., malformed request syntax, invalid request message framing, or deceptive request routing).

401 Unauthorized

Although the HTTP standard specifies "unauthorized", semantically this response means "unauthenticated". That is, the client must authenticate itself to get the requested response.

402 Payment Required

The initial purpose of this code was for digital payment systems, however this status code is rarely used and no standard convention exists.

403 Forbidden

The client does not have access rights to the content; that is, it is unauthorized, so the server is refusing to give the requested resource. Unlike 401 Unauthorized, the client's identity is known to the server.

404 Not Found

The server cannot find the requested resource. In the browser, this means the URL is not recognized. In an API, this can also mean that the endpoint is valid but the resource itself does not exist. Servers may also send this response instead of 403 Forbidden to hide the existence of a resource from an unauthorized client. This response code is probably the most well known due to its frequent occurrence on the web.

HTTP headers let the client and the server pass additional information with a message in a request or response. In HTTP/1.X, a header is a case-insensitive name followed by a colon, then optional whitespace which will be ignored, and finally by its value (for example: Allow: POST). In HTTP/2 and above, headers are displayed in lowercase when viewed in developer tools (accept: */*), and prefixed with a colon for a special group of pseudo-headers (:status: 200)

HTTP Header Types

Headers can be grouped according to their contexts:

Request headers

Contain more information about the resource to be fetched, or about the client requesting the resource.

Response headers

Hold additional information about the response, like its location or about the server providing it.

Representation headers

Contain information about the body of the resource, like its MIME type, or encoding/compression applied.

Payload headers

Contain representation-independent information about payload data, including content length and the encoding used for transport.

A **request header** is an HTTP header that can be used in an HTTP request to provide information about the request context, so that the server can tailor the response. For example, the Accept-* headers indicate the allowed and preferred formats of the response. Other headers can be used to supply authentication credentials (e.g., Authorization), to control caching, or to get information about the user agent or referrer, etc.

Request Headers Examples

GET /home.html HTTP/1.1
Host: developer.mozilla.org
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10.9; rv:50.0) Gecko/20100101 Firefox/50.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate, br
Referer: https://developer.mozilla.org/testpage.html
Connection: keep-alive
Upgrade-Insecure-Requests: 1
If-Modified-Since: Mon, 18 Jul 2016 02:36:04 GMT
If-None-Match: "c561c68d0ba92bbeb8b0fff2a9199f722e3a621a"
Cache-Control: max-age=0

Response Header

1. A **response header** is an HTTP header that can be used in an HTTP response and that doesn't relate to the content of the message. Response headers, like Age, Location or Server are used to give a more detailed context of the response.
2. Not all headers appearing in a response are categorized as *response headers* by the specification. For example, the Content-Type header is a representation header indicating the original type of data in the body of the response message (prior to the encoding in the Content-Encoding representation header being applied). However, "conversationally" all headers are usually referred to as response headers in a response message.

Response Header Examples

200 OK
Access-Control-Allow-Origin: *
Connection: Keep-Alive
Content-Encoding: gzip
Content-Type: text/html; charset=utf-8
Date: Mon, 18 Jul 2016 16:06:00 GMT Etag: "c561c68d0ba92bbeb8b0f612a9199f722e3a621a"
Keep-Alive: timeout=5, max=997
Last-Modified: Mon, 18 Jul 2016 02:36:04 GMT
Server: Apache
Set-Cookie: my-key=my value; expires=Mon, 17-Jul-2017 16:06:00 GMT; Max-Age=31449600; Path=/; secure
Transfer-Encoding: chunked
Vary: Cookie, Accept-Encoding
X-Backend-Server: developer2.webapp.scl3.mozilla.com
X-Cache-Info: not cacheable; meta data too large
X-kuma-revision: 1085259
x-frame-options: DENY

Representation Headers

1. A **representation header** (or 'representation metadata') is an HTTP header that describes how to interpret the data contained in the message.
2. For example, the content in a particular message might be encoded for transport, the whole resource might be formatted as a particular media type such as XML, JSON, HTML or Markdown, localized to a particular written language or geographical region, and/or compressed using a particular algorithm. The representation headers allow the underlying data to be extracted and understood. The underlying resource is semantically the same in each case, but its representation is different.
3. Representation headers may be present in both HTTP request and response messages with various methods. If sent as a response to a HEAD request, they describe the body content representation that would be selected if the resource was requested with a GET request.

Representation Headers Example

Representation headers include:

Content-Length

Content-Range

Content-Type

Content-Encoding

Content-Location

Content-Language

Validators used in conditional requests, such as:

Last-Modified

ETag

Payload Headers

1. A payload header is an HTTP header that describes the payload information related to safe transport and reconstruction of the original resource representation, from one or more messages. This includes information like the length of the message payload, which part of the resource is carried in this payload (for a multi-part message), any encoding applied for transport, message integrity checks, etc.
2. Payload headers may be present in both HTTP request and response messages (i.e., in any message that is carrying payload data).
3. The payload headers include: **Content-Length, Content-Range, Trailer, and Transfer-Encoding.**

Thank you

ANY QUESTIONS?